Character navigation in dynamic environments based on optical flow

Axel López, François Chaumette, Eric Marchand, Julien Pettré

EUROGRAPHICS 2019 / P. Alliez and F. Pellacini
(Guest Editors)

Volume 38 (2019), Number 2

Character navigation in dynamic environments based on optical flow

Axel López, François Chaumette, Eric Marchand, Julien Pettré

Univ Rennes, Inria, CNRS, IRISA, France
axel.lopez-gandia@irisa.fr, francois.chaumette@inria.fr, eric.marchand@irisa.fr, julien.pettre@inria.fr

Abstract

Steering and navigation are important components of character animation systems to enable them to autonomously move in their environment. In this work, we propose a synthetic vision model that uses visual features to steer agents through dynamic environments. Our agents perceive optical flow resulting from their relative motion with the objects of the environment. The optical flow is then segmented and processed to extract visual features such as the focus of expansion and time-to-collision. Then, we establish the relations between these visual features and the agent motion, and use them to design a set of control functions which allow characters to perform object-dependent tasks, such as following, avoiding and reaching. Control functions are then combined to let characters perform more complex navigation tasks in dynamic environments, such as reaching a goal while avoiding multiple obstacles. Agent's motion is achieved by local minimization of these functions. We demonstrate the efficiency of our approach through a number of scenarios. Our work sets the basis for building a character animation system which imitates human sensorimotor actions. It opens new perspectives to achieve realistic simulation of human characters taking into account perceptual factors, such as the lighting conditions of the environment.

CCS Concepts

• ***Computing methodologies*** → ***Modeling and simulation; Model development and analysis;*** *Model verification and validation; Animation;*

1. Introduction

Autonomous navigation capacities are essential for virtual characters to find their way in their environment and to move without colliding static and moving obstacles. Producing visually convincing animation requires the characters to move along realistic global navigation trajectories. The problem received a lot of attention and effort has been put to imitate the way humans form locomotion trajectories. The problem is generally addressed as an online motion control problem: given a current state (position, velocity), define the control signal that will lead the character to a desired state (a distant location), under various constraints (e.g., collision avoidance) allowing the system to form feasible human-like trajectories.

Various approaches were explored; all solutions are based on geometrical relations between the characters and the environment. This means that the terms involved in the motion control laws are relative velocities, positions or orientations, etc. This includes previous vision-based approaches [OPOD10, DMCN*17], which have set the idea of using a virtual retina to acquire all the information required for navigation. Information is projected on the virtual retina according the most basic laws of optics, but the nature of this information is not related to vision. As for other microscopic approaches, distance and velocity terms are used, and those terms are evaluated by directly accessing the simulation database to retrieve objects features.

In contrast with previous work, our main challenge is to control characters navigation based on the visual perception of the environment, and more especially the apparent motion of objects in their visual field. To this end, we restrict our input control variables to the optical flow perceived by characters, i.e., the apparent motion of objects in the 2D image formed on their retina. The information conveyed by the optical flow is sufficient: to detect moving objects or groups of objects moving together; to estimate if objects is moving closer or further; to detect if they are on a collision course. These estimations are performed based on a limited number of visual features that serve as input variables for a set of control and navigation functions for characters. Our approach has several advantages compared to previous one. By using only visual information, and similar visual features than those used by real humans, this work is a new step forward towards simulating humans locomotion control based on vision. Also, our navigation technique could apply to any system equipped with visual perception capabilities, such as a robot equipped with a camera.

This paper introduces several contributions. The first one is to establish formal relations between character motion and a set of visual features (cf. Section 5): how do the visual features change when the character motion is adjusted? Those relations are required to design motion control laws based on these visual features, which is our second contribution. We propose a set of object-related motion control functions allowing characters to move and adjust their

locomotion relatively to objects (cf. Section 6): to follow one object, to avoid one object or to reach one object. Our third contribution is a set of higher level navigation controllers, which are built by composing different control functions (cf. Section 7). For example, goal-driven navigation in cluttered environments is achieved by combining a control function for object reaching and a second control function to avoid any visible obstacles. All these technical elements compose a complete navigation simulation framework.

2. Related Work

Steering methods received a lot of attention because of their crucial role in the animation of virtual characters. They are approached as an automatic control problem: how to control a character motion for reaching a desired state such as position in the environment given the current state of the character and the one of the environment. The simplest approach is to control character accelerations based on geometrical variables directly related to motion, such as distances, angles, relative motion, etc. Reynolds [Rey99] pioneered this approach by demonstrating how a character can perform most of navigation tasks in static or dynamic environments by combining elementary steering behaviors.

The case of highly dynamic environments has been more deeply explored by microscopic crowd simulators. Simulation agents move by combining the interactions they have with the environments as well as with the surrounding agents [MT01]. Plethora of solutions explore different control laws to propel agents, they however build on similar principles. Agents, represented with a punctual geometrical model, are moved while maintaining geometrical and kinematics relation with their environment. Agents may maintain distances by repulsing each other [HM95], [EB97], by moving with "collision-free velocities" [PPD07], [vdBLM08], [MHT11], by applying navigation rules [ST05], by locally optimizing time-to-collision [KSG14]. A large literature build on these approaches and explore the differences in the simulation results [WJGO*14]. Nevertheless, all these approaches remain quite far from the sensorimotor loops by which real humans perform navigation tasks, because agents dispose of an accurate estimate of all state variables, and by the nature of the state variables used.

In the seek for higher realism of generated trajectories, more accurate biomechanics human models were explored. Instead of a simple punctual model, simulations were based on models of bipedal locomotion. For example, Bruderlin [BC89] or Boulic [BR04] provided a procedural approach to the generation of human locomotion. Singh [SKRF11] used an inverse pendulum to generate footstep sequences for crowd agents. Locomotion trajectories can also derive from a concatenation of examples of motion captured locomotion cycles, opening a large piece of literature about motion graphs [KGP02] [LCR*02], how to construct global trajectories on the environment [SH] or how to deal with dynamic environments [LK]. Reuse of recorded interactions is more specific however [LCL07], [KGML12], [CC14].

Also seeking for more realism, instead of considering the mechanical aspect of human motion, one idea was to consider more accurately the perceptual system, which is the input of the human control of locomotion. Several techniques appeared in the literature to provide solution to perform navigation in an environment [RTT90], or in a crowd [OPOD10] [DMCN*17]. Tu and Terzopoulos [TT, TR95] handle the navigation of other species like fishes. These vision-based approaches only partly reproduce the role of the human visual perception. As described by Patla [Pat97], vision provides the required information about near and far environment that is required to regulate locomotion both at local and global levels. In this sense, previous works reproduce correctly the role of vision: information required to perform navigation (e.g., time, distances or angle derivatives values) are projected on a sort of virtual retina, which is the only input to control agents' motion. The main advantage of vision-based approaches is to implicitly consider some effect of perception on motion, such as the visibility of obstacles and goals or the relative importance of obstacles with respect to the surface they visually cover. However, none of these approaches attempt to reproduce the most basic role of the human perceptual system which is to process the visual input (images perceived by the human eyes) to extract this information.

The robotics community has explored the use of visual features in order to develop autonomous robots capable of navigating in various environments. Braillon et al [BPCL06] proposed an obstacle detection algorithm based on discrepancies of the theoretical optical flow map (due to the robot's own motion) and the actual perceived optical flow. Other approaches attempted to use optical flow itself in the control laws such as Souhila and Karim [SK07]. They showed how the focus of expansion and time-to-collision can be extracted from optical flow maps and used a flow balance strategy to navigate in an environment. Aerial robots have also used optical flow such as the model proposed by Zingg et al. [ZSWS10] where depth was estimated from flow maps. Our work differs from these in that we do not balance the optical flow and instead use the optical flow features and its properties in our control law to detect and avoid obstacles.

In this paper, our objective is to steer agents based on the most basic visual cues they can acquire from their surroundings. As a main difference with previous approaches, we fully base our agent control on the position of objects in the perceived image, as well as in the dense optical flow generated by the relative motion between the agent and the environment, whereas previous methods provided a higher-level information to agents not requiring image processing steps. The goal of such an approach, at long term, is to implicitly consider the effect of perception on the character navigation. Nowadays, existing approaches are not able to consider the effect of lighting condition or salience of obstacles because there is no effect of luminosity on the virtual perception. In this paper, our contributions are a clear step in this direction, by showing which visual features can directly be extracted by the flow of images perceived by the agents, and by showing how they can be used to steer them in static and dynamic environments.

Our approach introduces the use of optical flow in highly dynamic environments. There are a number of approaches for optical flow computation. Black [BA91] estimated optical flow by imposing constraints to the image motion and solving the problem using a stochastic approach. There are many benchmarks [BSL*11] to test modern optical flow algorithms although it is still an open field of research. Our approach has a similar objective than active vision in robotics, the goal of which is to control a robot motion from vi-

sion sensors (camera). There are many types of problems tackled through active vision [CLK11], including motion control [CH06]. However, the case of mobile robotics in highly dynamic environments has not attracted a lot of attention. In this sense, our work also contributes to the robotics field, as our steering methods are directly transferable to a robot equipped with a camera, assuming that an accurate estimate of the optical flow is available.

3. Overview

This section provides a global description of our framework to steer agents (characters) in dynamic environments.

3.1. Agent Model

In this section, we provide a brief description of the agent.

- **Visual Sensors:** Agents are equipped with a virtual camera at the front of their head (simulating eyes). This camera is capable of recording the optical flow and the depth of the scene perceived at each update. The camera is always aligned with the direction of motion of the agent. The optical flow and depth images are the only information that are provided to our algorithm regarding the perception of the environment.
- **Control Variables:** We consider an agent with two degrees of freedom: forward acceleration a_c and turning rate ω_{cy}. Acceleration allows the agent to move faster or slower and turning rate is the angular velocity of the agent around its vertical axis, which allows the agent to change its direction of motion. Lateral motion is not considered in this model.

3.2. Algorithm Overview

Figure 1 shows the main components of the simulation loop by which each agents' motion is controlled at each time step:

- **Agent Perception:** in a first step, agents perceive the environment through a virtual camera set at the position of their head. From this point of view, we are interested in the apparent motion perceived on each pixel caused by the motion of obstacles relatively to the agent, i.e., the dense optical flow perceived by agents. We compute the dense optical flow for each agent synthetically, we remove its rotational component (i.e., we compensate for agent's head rotations). The computation of optical flow is detailed in Section 4. Additionally, agents record the depth in the scene corresponding to every pixel in the image.
- **Visual Features:** the next step is to extract some features from the perceived optical flow. First, the image is segmented using the optical flow image. This allows isolating obstacles with different apparent motion (most often corresponding to different close moving obstacles). As result of the segmentation a set of objects O_i, $i \in [0..n]$ (static or moving obstacles, other agents) are detected and their corresponding pixels x_j, $j \in [0..N]$ are extracted from the background. Then, per object O_i features are computed such as the position of the center g_{xi}, the Focus of Expansion (FOE) $\mathbf{f}_i$, time to collision τ_i and depth Z_i. The relevance and computation of these features are detailed in Section 5.

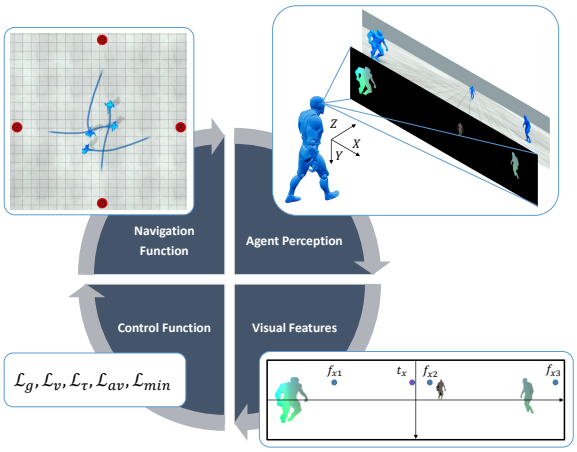


Figure 1: *Agent control loop scheme: every frame, each agent follows these steps to move in dynamic environments.*

- **Control Functions:** the third step is to evaluate a set of control functions $\mathcal{L}_k$ based on visual features and the agent's speed. Each control function relates agent's motion with a visual feature or its velocity. It is a basic element to build agents' visual-motor loops. For example, we define a control function $\mathcal{L}_t$ that controls the projection of geometrical points in the image which is used to perform goal reaching, this is detailed in Section 6. Other control functions may control the position of an object's focus of expansion $\mathbf{f}_i$ by changing their velocity to adjust the object's relative motion. The set of control functions and the set of visual features agents are able to control is presented in Section 6.
- **Navigation Functions:** the final step of the simulation loop updates agent velocity so as to locally minimize a navigation function. A navigation function allows an agent to achieve a high-level navigation task such as reaching a goal while avoiding all the obstacles on his way, the value of which reaches zero when the task is achieved at best. Each navigation function is a combination of control functions, i.e., a high level task is formulated as a set of visual features to control. This is achieved by formulating navigation functions as a combination of control behaviors: $\mathcal{L} = \sum \alpha_k \mathcal{L}_k$, where α_k weights the relative importance of each control function. This is explained in Section 7.

3.3. Mathematical Notation

In Table 1 the mathematical notations used through the paper are summarized. Through the paper all magnitudes are expressed in a coordinate frame with origin in the agent's camera, the z axis in the direction of the optical axis, y points towards the ground and x towards the right of the agent.

The coordinates of 3D and 2D points are noted as $\mathbf{X}_j = (X_j, Y_j, Z_j)$ and $\mathbf{x}_j = (x_j, y_j)$. Optical flow is noted as $\mathbf{u}_j = (u_j, v_j)$. All image points are expressed in normalized coordinates. Subindex c correspond to magnitudes specific to the agent.

Symbol	Description
$\mathbf{x}_j$	Position of pixel j in the image (2D point)
$\mathbf{X}_j$	Position of point j in the 3D world relative to the agent (3D point, m)
$\mathbf{u}_j$	Optical flow of pixel j (2D vector)
$\mathbf{f}_i$	Focus of Expansion (FOE) of object i (2D point)
$\mathbf{g}_i$	Position of the center of object i in the image (2D point)
$\mathbf{t}$	Position of the target (goal) in the image
$\mathbf{v}_i$	Velocity of obstacle i relative to the agent (m/s)
v_c	Speed of the agent (m/s)
a_c	Acceleration of the agent (m/s^2)
ω_{cy}	Angular velocity of the agent (rad/s)
τ_i	Time-to-Collision with object i (s)
Z_i	Average depth of object i with respect to the agent (m)

Table 1: *Description of symbols.*

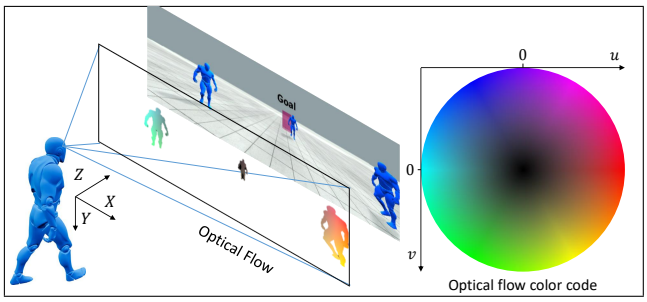


Figure 2: *Agents are equipped with cameras capable of recording rotationless optical flow at each frame. Color coding of the optical flow map* (u_j, v_j).

4. Agent Perception

In this section, we show how optical flow is computed. Optical flow is defined as the apparent motion of objects in the visual field. For the modeling, we approximate it with the projected motion in the image plane. A dynamic observer with angular velocity ω_{cy} perceives a point $\mathbf{X}_j$ of an object i, with relative translational velocity $\mathbf{v}_i$. We neglect optical flow components due to the object's own rotation under the assumption that these rotations are small compared the other contributions. The observer will perceive optical flow [SBC96] according to the following equation,

$$\begin{cases} u_j = \left(v_{ix} - x_j v_{iz}\right)/Z_j - \omega_{cy}\left(x_j^2 + 1\right) \\ v_j = \left(v_{iy} - y_j v_{iz}\right)/Z_j - \omega_{cy} x_j y_j. \end{cases} \quad (1)$$

with $(x_j, y_j) = (X_j/Z_j, Y_j/Z_j)$ being the projection of $\mathbf{X}_j$ in the image plane.

Equation (1) takes into account the angular velocity of the observer. We need to remove this contribution from the perceived flow. The reason is twofold: first, humans are able to compensate rotational component through the Vestibulo-ocular reflex [Ang04], and second, it allows us to define the FOE and other features derived from it. Then, Equation (1) is reduced to the following expression,

$$\begin{cases} u_j = \left(v_{ix} - x_j v_{iz}\right)/Z_j, \\ v_j = \left(v_{iy} - y_j v_{iz}\right)/Z_j. \end{cases} \quad (2)$$

Knowing the angular velocity of the agent, the optical flow map can be compensated to remove the rotational contribution. Figure 2 illustrates the perception of an agent and the colors of optical flow maps.

5. Visual Features

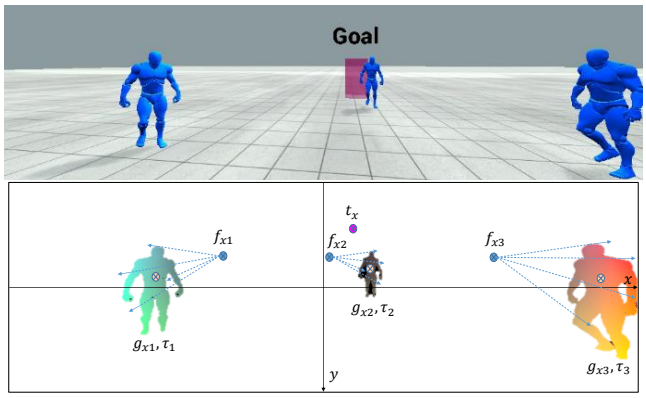


Figure 3: *Segmentation of the optical flow image from Figure 2. (top) Color image perceived by the agent. (down) Visual features extracted from optical flow. The goal* $\mathbf{t}$ *of the agent corresponds to the pink point. For every object the visual features extracted are the FOE* $\mathbf{f}_i$ *(blue points), time-to-collision* τ_i*, object center* $\mathbf{g}_i$ *(orange points) and optical flow vectors (blue arrows), which exhibit a radial configuration around the FOE allowing its computation as the intersection of many lines. The pink object is a representation of the goal and does not generate any optical flow.*

In this section, we expose the relevant features encoded in an optical flow image and show their dependency with the control variables. The relation between the translational velocity and the control scheme is given by $\dot{v}_{iz} = -a_c$ and $\dot{v}_{ix} = -v_c \omega_{cy}$. All features presented in this section are object specific. As explained in Section 3, objects are extracted from the background in the flow image. The first step is to remove the flow generated by the ground. We assume a flat ground and knowing the agent's height h, the depth of the points belonging to the ground are given by $Z(x_j, y_j) = h/y_j$ and the flow as $(u_j, v_j) = (x_j y_j, y_j^2) v_c/h$. Pixels matching this depth or flow values are removed. Then, we use the OpenCV library to detect discontinuities in the flow map through the Graph Segmentation algorithm [FH04]. The segmentation results in various objects, which are treated as rigid bodies. Figure 3 illustrates the various features contained in the optical flow image. Finally note that, as the motion of our agents is restricted to the horizontal plane, we focus on the x component of the features. Following figures displays visual features at the center of the vertical axis for this reason.

5.1. Focus of Expansion

The FOE is a per-object feature. It is defined as a point with null optical flow despite having a non-zero 3D velocity. It corresponds to a point that moves towards or away from the observer. Therefore, controlling the FOE allows us to prevent collisions. From (2) and enforcing $\mathbf{u}_j = \mathbf{0}$ and $\|\mathbf{v}_i\| \neq 0$ the FOE is expressed as,

$$\begin{cases} f_{xi} = v_{ix}/v_{iz}, \\ f_{yi} = v_{iy}/v_{iz}, \end{cases} \tag{3}$$

which is the projection of the relative velocity of the object in the image.

Optical flow vectors exhibit a radial configuration around this point, which allows the FOE to be computed as the intersection of many lines. These lines are defined as supporting point $\mathbf{x}_j$ and direction vector $\mathbf{u}_j$. Figure 3 illustrates the flow vectors configuration.

Equation (3) can be derived to obtain the time variation of the FOE with respect to the control variables. This results in the following expression,

$$\dot{f}_{xi} = \left(\frac{v_c}{v_{iz}} - f_{xi}^2 - 1 \right) \omega_{cy} - \frac{f_{xi}}{v_{iz}} a_c. \tag{4}$$

5.2. Time-to-collision

The time-to-collision is defined as the remaining time until an object will cross the plane perpendicular to the direction of motion of the agent. It is expressed as $\tau_i = -\frac{Z_i}{v_{iz}}$. From the FOE and the optical flow it is possible to compute the time-to-collision [TGS91] without any 3D geometrical information nor 3D velocity. For each pixel of the object we define its distance to the FOE as $\Delta_j \equiv \|\mathbf{x}_j - \mathbf{f}_i\|$ and we obtain

$$\tau_i = \sum_{j \in i} \frac{\Delta_j}{\|\mathbf{u}_j\|}. \tag{5}$$

It should be noted that although Δ_j and $\mathbf{u}_j$ are pixel dependent values, τ_i is not. When noise is present, in order to obtain a robust result, τ_i is computed as the average for all pixels of the object as expressed in (5). Finally, τ_i allows us to express the relative velocity of an agent in terms of the known information, $\mathbf{v}_i = -(f_{xi}, f_{yi}, 1)\tau_i/Z_i$.

5.3. Dynamics of Optical Flow

Finally, optical flow is used as a visual feature as well. The temporal derivative of (2) allows to obtaining the following expression,

$$\dot{u}_j = \frac{2}{\tau_i} u_j - \frac{v_c}{Z_j} \omega_{cy} + \frac{x_j}{Z_j} a_c. \tag{6}$$

This will be used to perform velocity alignment and following.

6. Control Functions

In this section we present a set of control functions $\mathcal{L}$ based on the visual features presented in the previous section. These functions are used to perform various tasks that allow an agent to navigate in a virtual environment. Therefore, we can update the control variables in order to minimize the value of the control functions.

6.1. Target Reaching

Control of geometrical points is a well-known problem in robotics [CH06]. It allows us to place a point (the goal) in a specific location in the image. In our framework, we want the goal t_x to be placed at the center of the image. This will cause the agent to move towards it. We use the following control function,

$$\mathcal{L}_t = \frac{1}{2} t_x^2, \tag{7}$$

The minimum value of this function is for $t_x = 0$ and will prevent the agent from deviating too much from the goal. The goal is a known point in space and in the image. The goal corresponds to a 3D point in space, therefore it follows (1) and its variation with the control variables is,

$$\dot{t}_x = t_x/\tau_t - (t_x^2 + 1)\omega_{cy}. \tag{8}$$

In the cases where the goal is a region and not a point, t_x is considered as the closest point of the region to the agent.

6.2. Collision Avoidance

As explained in Section 5.1 the FOE is the key element in preventing collisions. The geometrical point of an object that coincides with the FOE moves towards the agent resulting in a future collision. Therefore, the FOE needs to be displaced away from the object to prevent a collision. Additionally we need to take into account the size of the agent to prevent collision not only with the camera but also with the rest of the body. The control function needs to produce high values when the FOE f_{xi} is over the object or close to g_{xi}. To this end, we propose the following cost function,

$$\mathcal{L}_{av} = \sum_i^n I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i} \right). \tag{9}$$

where $\Delta x_i = g_{xi} - f_{xi}$, $\sigma_i = \frac{w_i}{ln(10)}$, being w_i the width of the object plus the width of the agent projected to the image, this is illustrated in Figure 4. $I(\tau_i) = a\tau_i + b$ is a linear function introduced to give a higher value to objects with lower time-to-collision τ_i. In our implementation we set $a = -2$ and $b = 10$ and the contribution of objects with $\tau_i > 5s$ is clamped to 0.

$\mathcal{L}_{av}$ takes into account all objects that need to be avoided. The goal of this function is to separate every FOE from its respective object to obtain a collision free trajectory.

From Equations (1) and (4), Δx_i has the following dependency with the control variables,

$$\dot{\Delta x}_i = (g_{xi} - f_{xi})/\tau_i + \left(f_{xi}^2 - g_{xi}^2 - \frac{\tau_i}{Z_i} v_c \right) \omega_{cy} + \frac{\tau_i}{Z_i} f_{xi} a_c. \tag{10}$$

6.3. Velocity Alignment

Velocity alignment consists in matching the direction of motion of the agent with a certain object. Velocity alignment is modeled as the minimization of the optical flow,

$$\mathcal{L}_{min} = \frac{1}{N^2} \left(\sum_{j \in i} u_j \right)^2, \tag{11}$$

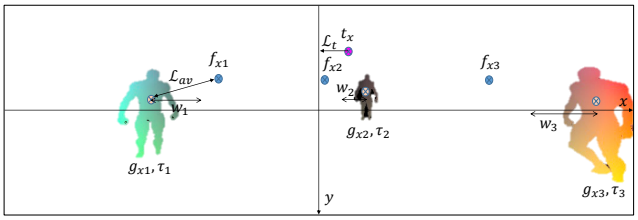


Figure 4: *Control Functions. Each control function has a different purpose. $\mathcal{L}_{av}$ tries to separate the FOE f_{xi} and the center of the object by more than w_i. w_i is the size of the object plus the size of the camera. $\mathcal{L}_t$ tries to place the goal t_x to the center of the image.*

being N the number of pixels of object i.

This function is intended to model a following behavior in conjunction with (13). The relation of u_j with the control variables has been presented in (6).

6.4. Speed Control

Some control functions may change the speed of the agent without any limit on their own. To control this behavior a preferred speed v_c^* is defined, that will play the role of the default speed when no other behavior is active and will limit the range of possible values it may take. This is modeled by,

$$\mathcal{L}_v = \frac{1}{2}(v_c - v_c^*)^2, \tag{12}$$

and from the definition of v_c, $\dot{v_c} = a_c$.

Alternatively, we may want to match the relative speed $v_{iz} = \frac{Z_i}{\tau_i}$ between the camera and an object. This is for example useful to follow an object. To this end, we will use,

$$\mathcal{L}_\tau = \frac{1}{2}v_{iz}^2, \tag{13}$$

with $\dot{v}_{iz} = -a_c$.

7. Navigation Functions

In this section we present how higher-level navigation tasks can be achieved by combining basic control functions. The key idea is to blend these functions to generate a navigation function $\mathcal{L}$, which capture a complex behaviour, such as reaching a target while avoiding collisions with multiple objects, or following a mobile target while avoiding collision with obstacles. The general expression of a navigation function is,

$$\mathcal{L} = \sum_k \alpha_k \mathcal{L}_k, \tag{14}$$

where $\mathcal{L}_k$ is a control function presented in Section 6 and α_k is a parameter that controls the weight of each control function.

Our control variables are the agent's acceleration a_c and angular velocity ω_{cy} around the vertical axis. At every update step, the gradient of the navigation function is evaluated and the control variables are updated by,

$$a_c = -\lambda \frac{\partial \mathcal{L}}{\partial v_c}, \; \omega_{cy} = -\lambda \frac{\partial \mathcal{L}}{\partial \theta}, \tag{15}$$

with λ being the gradient descent step size parameter. We have empirically found that a value $\lambda = 1$ performs well.

As an example, thanks to (8) using $\mathcal{L}_t$ from (7) results in the control variables taking the following values, $a_c = 0$, $\omega_{cy} = -(1+t_x^2)t_x$.

8. Results

In this section we propose two navigation functions that follow the scheme presented in Section 7 and present the trajectories generated by our algorithm when agents face different scenarios. In the two following sections, we demonstrate that the few visual features extracted from the perceived optical flow that compose our control functions are sufficient to steer characters in high level navigation tasks such as reaching destinations in dynamic environments, or following moving objects. The extraction of visual features is however performed at a computational costs, that we evaluate in Section 8.3.

8.1. Collision-free goal reaching navigation

We first demonstrate our technique in the classic situation of a character reaching a goal while avoiding the obstacles of a given environment. To model this behavior we build a navigation function combining Equations (7), (12) and (9). The navigation function is as follows,

$$\mathcal{L} = \alpha\mathcal{L}_t + \beta\mathcal{L}_v + \gamma\mathcal{L}_{av}, \tag{16}$$

We want to drive the goal to the center of the image to make sure we move towards it and the comfort velocity $v_c^* = 0.15m.s^{-1}$. In the following examples we will use $\alpha = 1, \beta = 0.1, \gamma = 1$ unless otherwise specified. In collision avoidance tasks the system tends to apply a very small variation in acceleration therefore the value of β affects only how fast the speed of the agent reaches the desired speed v_c^*. Empirically $\beta = 0.1$ yields good results. We will later show the effect of the relative weight of α and γ.

The gradient in (15) is computed for (16) using (8) and (10), leading the control variables to have the following expression,

$$\begin{aligned} a_c &= -\lambda \Big[\beta(v_c - v_c^*) + \\ &\quad \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \frac{-f_{xi}\tau_i}{Z_i}\Big], \\ \omega_{cy} &= -\lambda \Big[-\alpha t_x(t_x^2+1) + \\ &\quad \gamma \sum_i I(\tau_i) \exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right) \frac{sign(\Delta x_i)}{\sigma_i} \left(f_{xi}^2 - g_{xi}^2 - \frac{\tau_i}{Z_i}v_c\right)\Big], \end{aligned} \tag{17}$$

which will update the simulation closing the control loop in Figure 1. We demonstrate this navigation function in static and dynamic environments.

8.1.1. Static Environments

In a static environment, the apparent motion of objects is only due to the agent's own motion. This causes the FOE to be at the center of the image at all times $\left(\frac{\partial f_{xi}}{\partial \theta} = 0, \frac{\partial f_{xi}}{\partial v_c} = 0\right)$.

Figure 5 shows a complex room with objects of various sizes and shapes. Our vision-based algorithm is capable to avoid any object to cross the room. Each agent takes a different route depending on its initial position because of the local optimization. As x_{g3} is farther away it has a greater τ_i than other objects and therefore its contribution to $\mathcal{L}_{av}$ is lower. Therefore, the reaction to it takes place later in the simulation.

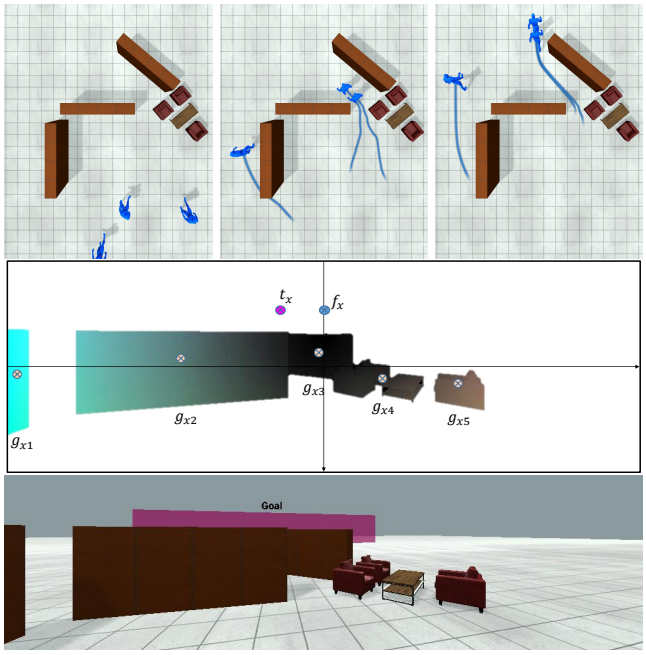


Figure 5: *Agent trajectory with static obstacles. (up) Top view. Three agents need to cross the room avoiding obstacles. Various objects of different shapes and sizes are present in this scene. Agents are placed in different starting position and orientation and they choose different paths to walk out the room. (middle) Visual features perceived by one of the agents. (down) Color image perceived corresponding to the visual features.*

Figure 6 shows an example of an agent in a city-like environment. Due to using the entire optical flow image the agent is capable of avoiding objects with complex geometry like the fence and of other shapes such as the tree.

8.1.2. Dynamic Environments

Now we let agents face other agents using the same navigation function. Therefore, the FOE of every object will depend on the motion of the corresponding agent.

Figure 7 shows a circle scenario. Six agents are arranged in a circular formation and are tasked to reach the opposite side of the circle. The symmetry of this scenario causes agents to follow the same pattern.

Figure 6: *Agent trajectory in a city-like environment with urban obstacles.*

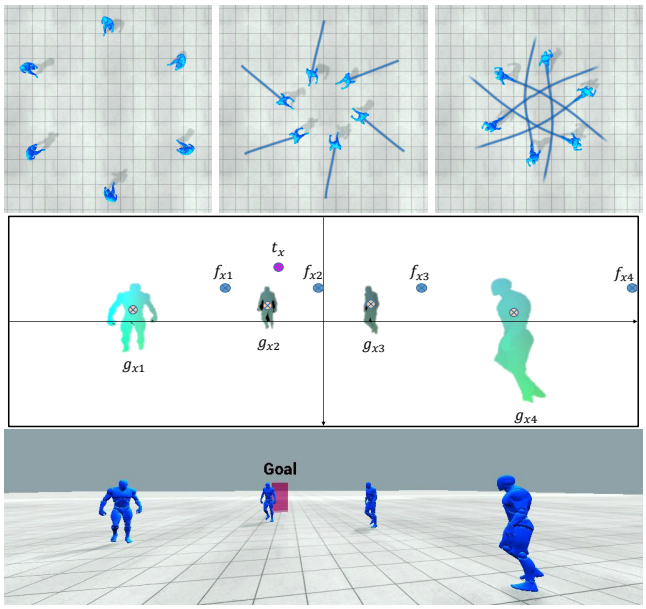


Figure 7: *Circle scenario. (up) Six agents are initially arranged in circular formation and need to reach opposite side of the circle. (middle) Visual features perceived by one agent. (down) Color image corresponding to the same agent.*

Figure 8 shows a scene of multiple agents in a street with cars. Agents can adapt their paths to avoid both types of obstacles (cars and other agents) and are capable of crossing the street without collision.

Figure 9 shows two groups of agents tasked to reach the opposite side of the room. This scenario shows the effect of different values for γ. When it is too low (b) the agents barely adapt to other agents and some collisions occur. As γ becomes greater agents with the same goal form groups and increases the distance between them.

Figure 10 shows how the cost function and goal of an agent varies over time. The cost of avoidance increases and decreases very fast as objects enter and leave the field of view. Once all obstacles are cleared goal reaching becomes dominant.

The supplementary video contains additional examples to further illustrate our algorithm. We show examples of agents in various scenarios avoiding static and dynamic obstacles and walking in restricted spaces.

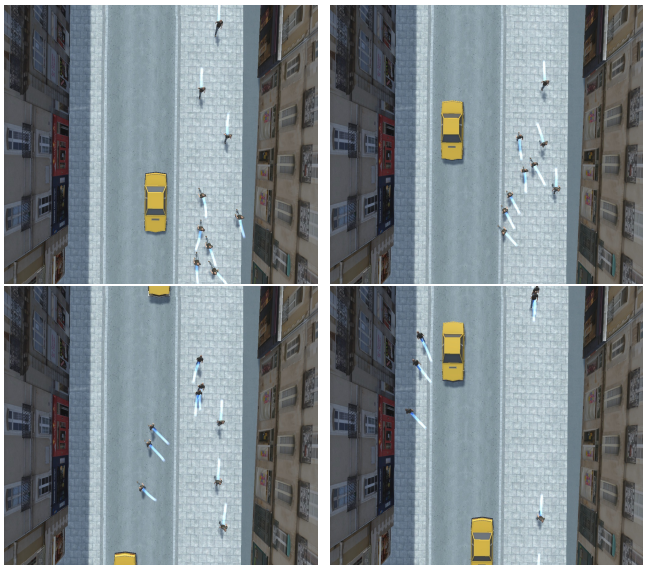


Figure 8: *Agent trajectory with complex obstacles in a city-like environment.*

8.2. Following

The second navigation function involves following. We build a navigation function that models an agent following an object or another agent. Therefore, our navigation function takes into account velocity alignment and speed matching. The agents need to be able to avoid obstacles as well. Equations (11), (13) and (9) are thus combined resulting in the following expression,

$$\mathcal{L} = \alpha\mathcal{L}_{min} + \beta\mathcal{L}_{\tau} + \gamma\mathcal{L}_{av}, \tag{18}$$

$\mathcal{L}_{min}$ controls the direction of motion of the agent and $\mathcal{L}_{\tau}$ controls its speed.

Agents are assigned a leader and they are tasked to follow it. Agents are drawn in blue while leaders are drawn with red color. The parameters used in this section are $\alpha = 1, \beta = 1, \gamma = 1$. In this situation β affects how fast a blue agent adapts to the variation of the speed of a red agent.

Similarly as we did for (17), we derive the expression of the control variables from the navigation function in (18),

$$\begin{aligned} a_c &= -\lambda\left[-\beta\frac{Z_{red}}{\tau_{red}} + \right. \\ &\left. \gamma\sum_i I(\tau_i)\exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right)\frac{sign(\Delta x_i)}{\sigma_i}\frac{-f_{xi}\tau_i}{Z_i}\right], \\ \omega_{cy} &= -\lambda\left[-\alpha\left(\frac{2v_c}{NZ_{red}}\sum_{j\in red}u_j\right) + \right. \\ &\left. \gamma\sum_i I(\tau_i)\exp\left(-\frac{|\Delta x_i|}{\sigma_i}\right)\frac{sign(\Delta x_i)}{\sigma_i}\left(f_{xi}^2 - g_{xi}^2 - \frac{\tau_i}{Z_i}v_c\right)\right], \end{aligned} \tag{19}$$

to update the simulation.

Figure 11 shows an agent following another one. The red agent goes through a set of way-points. Every time the red agent changes its direction to reach the next way-point the blue agent adapts its velocity to align with the one of the red agent. For higher values of α the adaptation is faster and the trajectory of the blue agent becomes similar to the trajectory of the red agent. For lower values of α the adaptation is much slower producing a different trajectory.

Figure 12 shows a situation where obstacles are in the way of the agents while following another agent. Depending on the initial conditions the agents choose different paths to perform following and avoidance.

8.3. Performance

In this section, we give indicative numbers about our approach performance. Simulations ran on a 2.17GHz Intel Core i7 processor, 16 GB of RAM, Nvidia Quadro M2000M using Unreal Engine 4 as the graphics engine.

The performance of our algorithm depends on the chosen resolution of the images. The GPU renders the flow image perceived by the agent and then they are downloaded to the CPU. The data transfer consumes a lot of time and thus we believe performance could be greatly improved with a full GPU implementation. We ran most of the simulations using a high-resolution camera 1024x1024. Under these conditions, every agent takes on average 0.48s to update. We can lower the resolution to 300x300 increasing performance to 0.03s per agent. Figure 13 shows the same scenarios shown through the paper with high and low flow resolution. High-resolution flow results in a quite straight trajectory while low resolution flows produces trajectories with higher curvature and noise due to the reduced precision in object position. In the first comparison, an agent even takes a very different path.

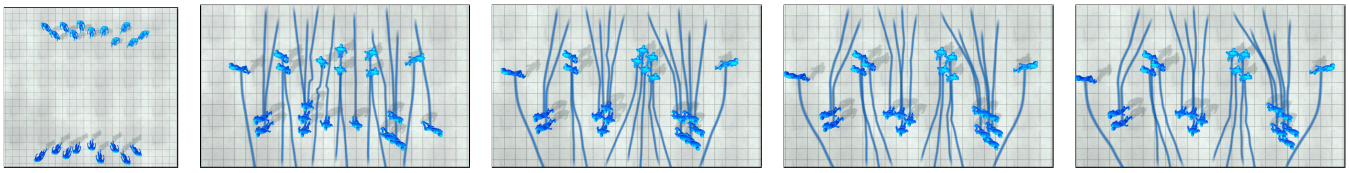
Figure 14 shows a comparison of trajectories of the scenario shown in Figure 9 for different resolutions and the computation time per-agent and per frame. For lower resolutions, collisions occur more often due to the reduced spatial precision. In addition, low-resolution images causes the object segmentation to be less accurate.

9. Discussion

The previous section demonstrates our steering approach in action. It demonstrates that using optical flow features, we are able to steer agents so that they perform navigation tasks, such as goal-directed navigation among static and dynamic obstacles. In this section, we discuss our results as well as the limitations of our approach.

9.1. Level of realism and comparison with previous approaches

Our objective is to reconsider the nature of inputs used for trajectory control, with the aim of simulating more accurately the low level processes by which human use vision to navigate. We demonstrate that few visual features are enough to steer characters in goal-directed navigation tasks performed in dynamic environments. Because some crowd simulation algorithms address similar problems, we compare some of them with our approach.



(a) *Initial position.* **(b)** $\gamma = 0.1$ **(c)** $\gamma = 0.5$ **(d)** $\gamma = 1.0$ **(e)** $\gamma = 1.5$

Figure 9: *Two rows of agents navigate towards the opposite side of the room avoiding each other. As γ increases, the distance between agent with different goals increases.*

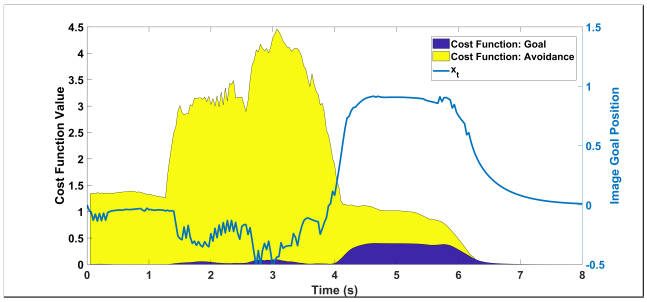


Figure 10: *Variation of the cost function (left axis) and goal t_x (right axis) over time. It corresponds to the front agent of the scenario shown in Figure 5.*

Beyond the nature of input variables, the principles set by our control scheme have similarities with previous approaches: as [KSG14, DMCN*17] we locally optimize a navigation function to control instantaneous velocity and acceleration. Also, our navigation functions have similarities with previous approaches. For example, avoidance is based on the estimation of the risk of future collision. In this sense, we are similar to the general principle of velocity-based approaches [PPD07, vdBLM08] and their many variants. Therefore, the crucial difference is the nature of variables to estimate the risk of collision. Velocity based approaches explicitly compute the future distance of closest approach. We get this estimation through the distance-between the visual center of an object with the focus of expansion of the flow generated by this object. Whilst resulting trajectories may exhibit similar properties (e.g., anticipated collision avoidance), they will not be strictly identical. We believe our system could be used to identify human sensorimotor functions that link vision and locomotion. This comparison however needs large efforts, which would first consist in acquiring visuomotor data to describe real human behaviors.

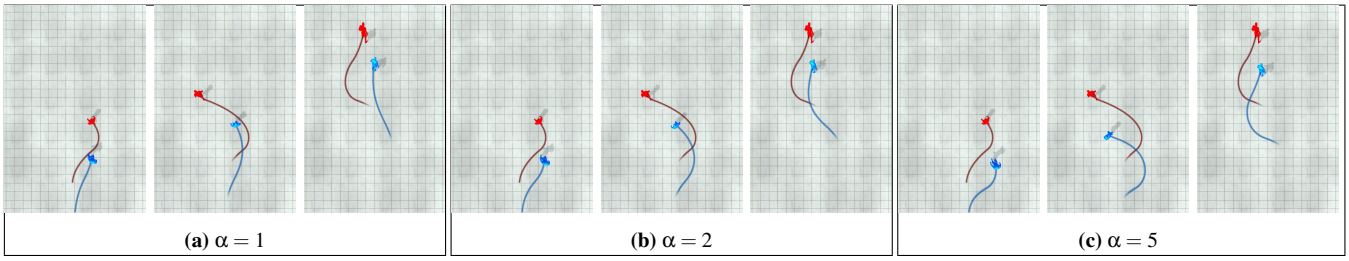
We now provide a qualitative comparison of our algorithm, and more specifically the collision avoidance function (Eq. 16), with RVO2 [vdBGLM11] and the model proposed by Dutra et al [DMCN*17]. We chose those two approaches because they are good representative of two categories of approaches: velocity-based and vision-based approach. These algorithms share many similarities with ours. All of them evaluate the future risk of collision between the character and surrounding obstacles and adjust the characters motion accordingly. Their difference resides in the way this risk is evaluated, and motion is adjusted. Obviously, the main difference is that our approach relies on the evaluation of visual features whilst previous approaches exploit geometrical relations. The following comparison highlights the difference in the generated trajectories over the example of crossing groups.

Figure 15 shows two groups of agents crossing each other to reach the opposite side of the room with various initial densities. We compare our model with the aforementioned ones. RVO2 works well in low-density scenarios however, the distance between agents of different groups is just the minimum to prevent collisions. In Dutra's model and our model, the minimum distance between agents is not fixed but rather it adapts to the density of the crowd. Dutra's model also favors many variations of speed for avoidance while in our algorithm avoidance favors a change in orientation rather than a change in the speed of an agent due to the nature of the visual feature. Our model favors following groups of agents with the same direction while in other approaches agents may follow very different paths from one another. RVO2 presents congestion issues when the initial agent-to-agent spacing is much smaller than the size of the agents. Dutra's model and ours allows a continuous rearrangement of the formation over time to minimize the congestion issue.

Figure 16 shows a comparison of the angular velocity of the agents. Our model favors small progressive adaptations, Dutra's model favors sudden adaptations with large angular velocity and RVO favors large and small adaptations alike.

9.2. Limitations and Future Work

One of the motivations in our work is to consider better the effect of visual perception on motion control. Currently the optical flow map does not consider lighting conditions however, objects that lack contrast with the background or texture, or objects that are not correctly lighted would not generate optical flow, and this lack of information would implicitly impact motion control. We still need to introduce the notion of lack of information in the control loops. When the absence of optical flow is not due to the absence of near moving objects, but the impossibility to perceive them, reaction is different (e.g., humans avoid dark areas or are unable to move normally in the dark). To solve this problem we propose to change the way we compute optical flow. A numerical approach would take into account naturally the lightning conditions in the scene improving the capabilities of our framework. This poses the challenge of obtaining high enough quality flow maps since it is usually computationally expensive. We would like to evaluate the benefit of recent techniques based on deep-learning approaches [IMS*17] to obtain the optical flow map. We also would like to remove the need



(a) $\alpha = 1$ **(b)** $\alpha = 2$ **(c)** $\alpha = 5$

Figure 11: *Following a leader. The blue agent must follow the red one, which goes through a predefined set of way-points. We show the resulting trajectory for different values of* α.

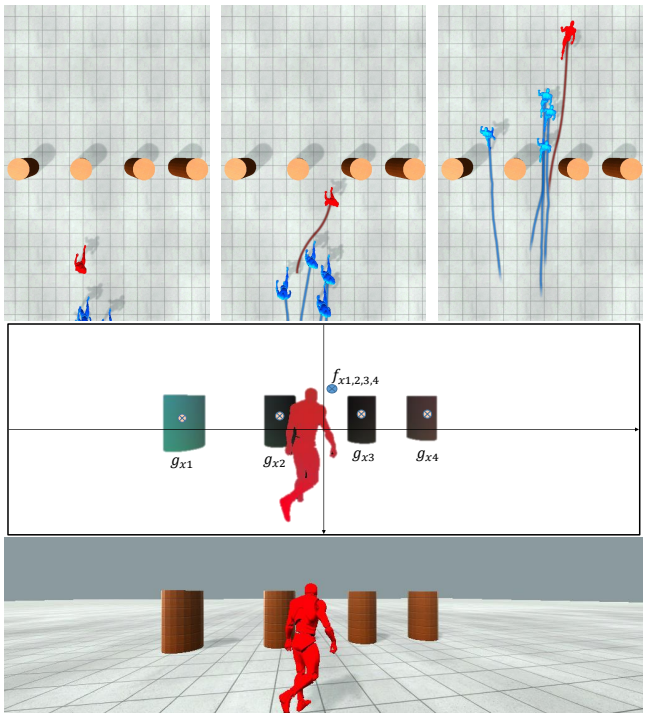


Figure 12: *Following scenario. (up) Four agents (blue) follow a leader (red) while avoiding obstacles. (middle) Visual features perceived by the agents. (down) Color image corresponding to the middle image.*

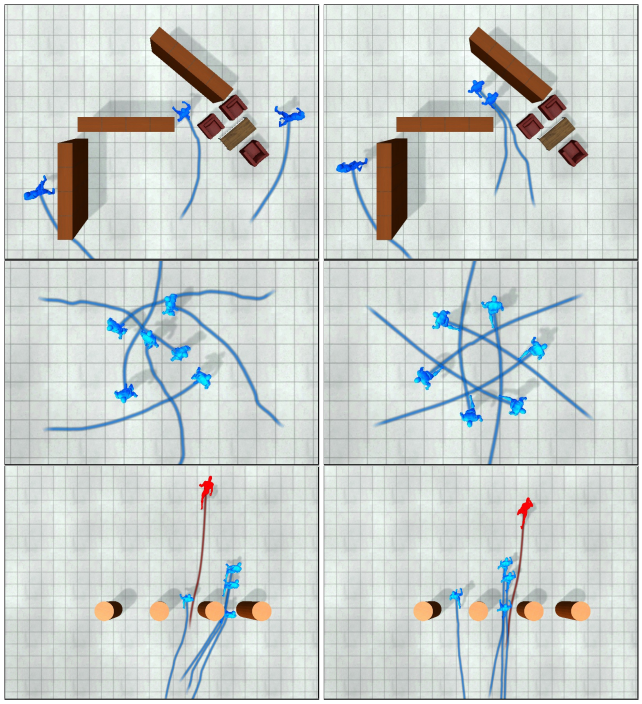
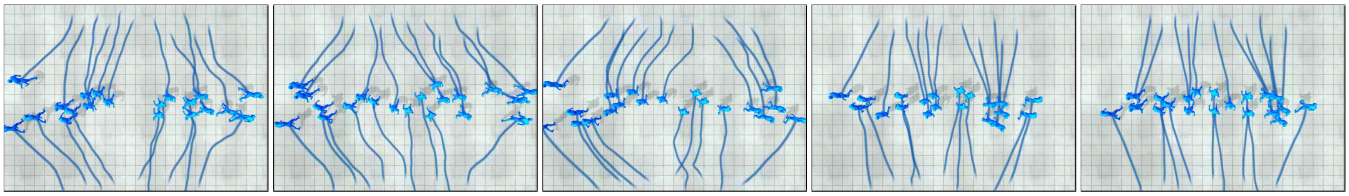


Figure 13: *Variation of trajectories depending on optical flow resolution. The left column shows the resulting trajectories of a 300x300 flow map resolution and the right column shows the same scenario with a 1024x1024 resolution.*

of depth information as an input to our algorithm. We believe that some assumptions can be made such as $\frac{\tau_i}{Z_i} v_c = \frac{1}{2}$ which assumes that objects are moving at the same velocity as the agent and yet avoidance would still be robust within a certain range of object velocities.

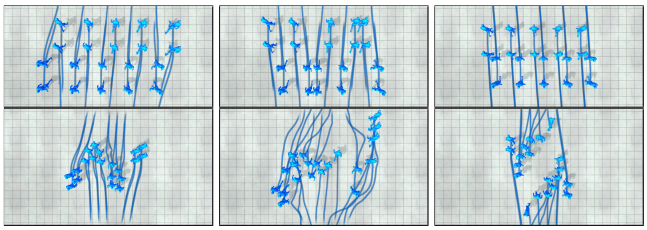
We have demonstrated how relatively simple navigation function perform well in different scenes, made of multiple static and dynamic obstacles. We have demonstrated that our method can combine following and avoidance behaviors for example. There are many other kinds of navigation tasks we would however like to explore, for example multiple target tracking, or object interception (catching), which have also been left unexplored by previous approaches. Our algorithm relies on a cost function optimization. Depending on the scenario, we may face the problem of local minimum. This may cause the agent to collide while trying to avoid multiple objects if there is not enough room for it go through. This can be avoided by considering two objects as a single one if they are too close in the image, or also by playing on parameters value (e.g., to diminish the importance of goal reaching and let the character exploring detours). While the meaning and the tuning of our parameters is straightforward, we need to more deeply explore how our navigation generalizes and how relative weights α and γ could be automatically adapted to the visual context.

Beyond the question of behaviors and parameters, we are also interested in applying our approach to simulate the collective motion of animal species, such as birds or fishes.



(a) *(64x64) 0.003s* **(b)** *(128x128) 0.006s* **(c)** *(256x256) 0.027s* **(d)** *(512x512) 0.111s* **(e)** *(1024x1024) 0.48s*

Figure 14: *Trajectories of the scenario shown in Figure 9 for various resolutions. Computation time is shown for every resolution.*



(a) *Our model* **(b)** *Dutra* **(c)** *RVO2*

Figure 15: *Lane formation. Agents are tasked to reach the opposite side of the room. (top) Agents are initially arranged in a low-density configuration, agent spacing is larger than agent size. (bottom) Agents are arranged in a high-density configuration, agent spacing is close to zero.*

Our method attempts to simulate some low-level sensorimotor mechanism of human locomotion. To address more complex scenarios, we need to consider the role of other components, such as for example visual attention, and how more generally use head motion to perform visual exploration of the environment (which would address issues due to the limited field of view of agents), or the coupling with basic motion behaviors with higher level ones such as planning.

10. Conclusions

We presented a new agent steering approach using synthetic vision. The main contribution of this paper is to formulate a set of steering behaviors that are directly expressed with the visual features related to the optical flow generated by the motion of the agent and dynamic obstacles. We built a set of navigation functions on top of these steering behaviors to let agent autonomously perform complex navigation tasks that combine avoidance of obstacles with goal reaching. We have demonstrated that we can reproduce some results obtained by previous approaches, while, in contrast with them, we do not exploit geometrical relations between characters and their environment. This is interesting for several reasons, including the potential to support future research directions as we discussed, such as for example the identification of human perception-action loops. Future work also aim at considering a new range visual factors which influence human navigation, such the environment lighting conditions. Also, our approach closes a gap between Computer Graphics and Robotics. As we base on visual features only, a robot equipped with a camera and optical flow computation capabilities could be directly steered by our technique, whereas the use of previous techniques would need to map the environment so as to establish the required geometrical relations.

References

[Ang04] ANGELAKI D. E.: Eyes on target: What neurons must do for the vestibuloocular reflex during linear motion. *Journal of Neurophysiology 92*, 1 (2004), 20–35. PMID: 15212435. 4

[BA91] BLACK M. J., ANANDAN P.: Robust dynamic motion estimation over time. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), pp. 296–302. 2

[BC89] BRUDERLIN A., CALVERT T. W.: Goal-directed, dynamic animation of human walking. *SIGGRAPH 23*, 3 (July 1989). 2

[BPCL06] BRAILLON C., PRADALIER C., CROWLEY J. L., LAUGIER C.: Real-time moving obstacle detection using optical flow models. In *2006 IEEE Intelligent Vehicles Symposium* (2006), pp. 466–471. 2

[BR04] BOULIC R. ULICNY B. T. D.: Versatile walk engine. *Journal of Game Development* (2004). 2

[BSL*11] BAKER S., SCHARSTEIN D., LEWIS J. P., ROTH S., BLACK M. J., SZELISKI R.: A database and evaluation methodology for optical flow. *Int. Journal of Computer Vision 92*, 1 (Mar 2011), 1–31. 2

[CC14] CHARALAMBOUS P., CHRYSANTHOU Y.: The pag crowd: A graph based approach for efficient data-driven crowd simulation. *Computer Graphics Forum 33*, 8 (2014). 2

[CH06] CHAUMETTE F., HUTCHINSON S.: Visual servo control. i. basic approaches. *IEEE Robotics Automation Magazine 13*, 4 (2006), 82–90. 3, 5

[CLK11] CHEN S., LI Y., KWOK N. M.: Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research 30*, 11 (2011), 1343–1377. 3

[DMCN*17] DUTRA T. B., MARQUES R., CAVALCANTE-NETO J., VIDAL C. A., PETTRÉ J.: Gradient-based steering for vision-based crowd simulation algorithms. *Computer Graphics Forum 36*, 2 (2017). 1, 2, 9

[EB97] ERIC BOUVIER EYAL COHEN L. N.: From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electronic Imaging 6* (1997), 6 – 6 – 14. 2

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2 (Sep 2004), 167–181. 4

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Phys. Rev. E 51* (May 1995), 4282–4286. 2

[IMS*17] ILG E., MAYER N., SAIKIA T., KEUPER M., DOSOVITSKIY A., BROX T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Jul 2017). 9

[KGML12] KIM S., GUY S. J., MANOCHA D., LIN M. C.: Interactive simulation of dynamic crowd behaviors using general adaptation syndrome theory. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), I3D '12. 2

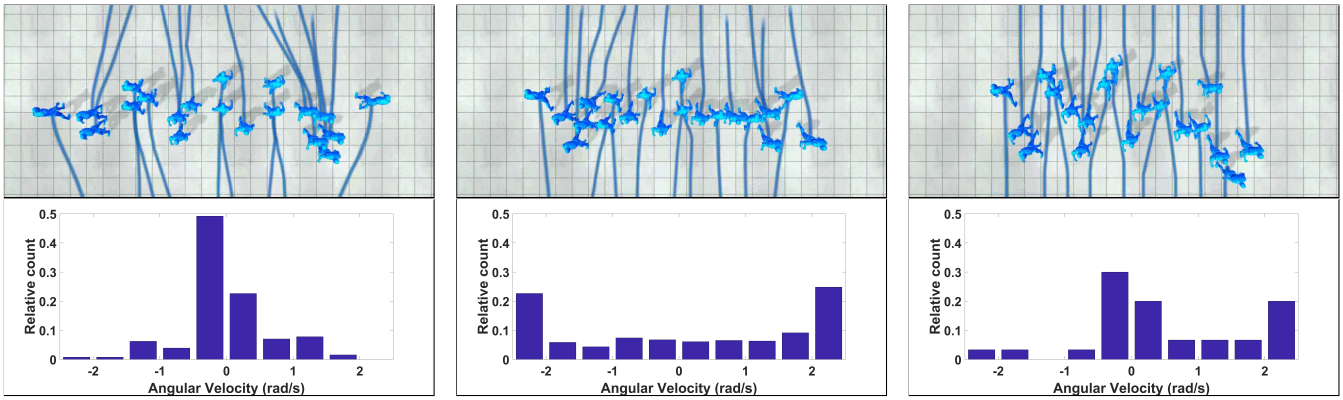


Figure 16: *Comparison of histograms of the angular velocity of an agent for different models: ours (left), Dutra (middle) and RVO (right). Only non-zero adaptations are being displayed.*

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. In *Proc. of the 29th Annual Conf. on Computer Graphics and Interactive Techniques* (2002), SIGGRAPH. 2

[KSG14] KARAMOUZAS I., SKINNER B., GUY S. J.: Universal power law governing pedestrian interactions. *Phys. Rev. Lett. 113* (Dec 2014), 238701. 2, 9

[LCL07] LERNER A., CHRYSANTHOU Y., LISCHINSKI D.: Crowds by example. *Computer Graphics Forum 26*, 3 (2007). 2

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (2002), SIGGRAPH '02. 2

[LK] LAU M., KUFFNER J. J.: Precomputed search trees: Planning for interactive goal-driven animation. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2

[MHT11] MOUSSAÏD M., HELBING D., THERAULAZ G.: How simple rules determine pedestrian behavior and crowd disasters. 6884–6888. 2

[MT01] MUSSE S. R., THALMANN D.: Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (2001), 152–164. 2

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Trans. Graph. 29*, 4 (July 2010). 1, 2

[Pat97] PATLA A. E.: Understanding the roles of vision in the control of human locomotion. *Gait & Posture 5*, 1 (1997), 54 – 69. 2

[PPD07] PARIS S., PETTRÉ J., DONIKIAN S.: Pedestrian reactive navigation for crowd simulation: a predictive approach. *Computer Graphics Forum 26*, 3 (2007). 2, 9

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. *Game Developers Conference* (1999), 763–782. 2

[RTT90] RENAULT O., THALMANN N. M., THALMANN D.: A vision-based approach to behavioural animation. *The Journal of Visualization and Computer Animation 1*, 1 (1990). 2

[SBC96] SUNDARESWARAN V., BOUTHEMY P., CHAUMETTE F.: Exploiting image motion for active vision in a visual servoing framework. *Int. J. Robot. Res. 15*, 6 (Jun. 1996), 629–645. 4

[SH] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 Papers*. 2

[SK07] SOUHILA K., KARIM A.: Optical flow based robot obstacle avoidance. *Int. Journal of Advanced Robotic Systems 4*, 1 (2007), 2. 2

[SKRF11] SINGH S., KAPADIA M., REINMAN G., FALOUTSOS P.: Footstep navigation for dynamic crowds. *Computer Animation and Virtual Worlds 22*, 2-3 (2011), 151–158. 2

[ST05] SHAO W., TERZOPOULOS D.: Autonomous pedestrians. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), SCA '05. 2

[TGS91] TISTARELLI M., GROSSO E., SANDINI G.: Dynamic stereo in visual navigation. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), pp. 186–193. 5

[TR95] TERZOPOULOS D., RABIE T. F.: Animat vision: Active vision in artificial animals. In *Proc., Fifth Int. Conf. on Computer Vision* (1995), IEEE, pp. 801–808. 2

[TT] TU X., TERZOPOULOS D.: Artificial fishes: Physics, locomotion, perception, behavior. 2

[vdBGLM11] VAN DEN BERG J., GUY S. J., LIN M., MANOCHA D.: Reciprocal n-body collision avoidance. In *Robotics Research* (Berlin, Heidelberg, 2011), Pradalier C., Siegwart R., Hirzinger G., (Eds.), Springer Berlin Heidelberg, pp. 3–19. 9

[vdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE Int. Conf. on Robotics and Automation* (2008), pp. 1928–1935. 2, 9

[WJGO*14] WOLINSKI D., J GUY S., OLIVIER A.-H., LIN M., MANOCHA D., PETTRÉ J.: Parameter estimation and comparative evaluation of crowd simulations. In *Computer Graphics Forum* (2014), vol. 33, Wiley Online Library, pp. 303–312. 2

[ZSWS10] ZINGG S., SCARAMUZZA D., WEISS S., SIEGWART R.: Mav navigation through indoor corridors using optical flow. In *2010 IEEE Int. Conf. on Robotics and Automation* (2010), pp. 3361–3368. 2